A Learning Tool Visualizing the Algebraic Laws for Optimizing Query Execution

Elitsa Arsova, Silyan Arsov

***Abstract**: In this paper, we present an interactive learning tool that visualizes one and the same "question" to a database in different ways using equivalent expressions and keeping the same result. The tool shows time cost of equivalent expressions' execution that enables students to compare and choose the most optimum and efficient SQL statement. The algebraic laws about relational operations and their implementation into the system are given. Finally, test results of the tool are presented.*

***Key words:** Learning Tool, Databases, Query Optimization, Algebraic Laws about Relational Operations, Equivalent Expressions, Time Cost.*

INTRODUCTION

Query optimization is of great importance for the performance of a relational database, especially for the execution of complex SQL statements [3]. Database students should be familiar with algebraic laws for optimizing query execution and common strategies for improving the logical query plan, although different database management systems (DBMS) offer query optimizers that determine the best strategy for performing each query and choose, for example, whether or not to use indexes for a given query, and which join techniques to use when joining multiple tables.

A query in a declarative language can be executed using a number of different ways that can generate the same result. That is why, it is essential to find the "best" way that will cost minimum process time and will achieve efficiency and effectiveness on very large databases. Hence, query performance in relational database systems depends not only on the database structure, but also on the way in which the query is optimized. Various classes of syntactically equivalent SQL queries are shown in [4], each of which can exhibit substantial differences in data access depending on the characteristics of the query formulation and the success of the database query optimizer.

An interactive learning tool that enables students to optimize and re-organize query plan and to view estimated time for a query execution, will be very beneficial and useful on Database practices. In this paper, we will present the importance of such educational tool for us and will expose the implementation process of it.

In recent years a small number of web-based tools have been proposed to help students learn to write SQL query statements and also to assess students' SQL writing skills. One of them is SQLify - a new SQL teaching and assessment tool that extends the current state-of-the-art by incorporating peer review and enhanced automatic assessment based on database theory to produce more comprehensive feedback to students [2]. Similarly, SQLator has an evaluate function, which allows a user to evaluate the correctness of his/her query formulation. The evaluate engine is based on complex heuristic algorithms. The tool also provides instructors the facility to create and populate database schemas with an associated pool of SQL queries [6]. But not always these functions of SQL learning tools are enough. According to [1], one of the most important features desirable in such a system is its ability to perform query transformation. The use of a universal symbol and tree manipulation system to perform query translation, decomposition and optimization is described by researchers. The system also visualizes examples of transformation rules required to translate SQL expressions into equivalent QUEL expressions, decompose SQL expressions into parse trees and perform optimization of expressions based on relational algebra. The results of every important phase of the query transformation during its execution are interactively available to the system user.

The optimization process is transformation of SQL statements so that these complex expressions can be transformed into equivalent, but better performing, SQL statements. The knowledge of equivalent query expressions and optimization strategies is significant to database students for their future realization.

The reminder of the paper is organized as follows. The next section briefly introduces equivalent expressions. Section 3 outlines the common strategies for improving the logical query plan. Section 4 describes the functional capabilities of interactive learning tool and introduces the implemented web-based system. The last section reviews the contributions of the paper.

EQUIVALENT EXPRESSIONS

There are algebraic laws that turn one expression into an equivalent expression which may have a more efficient physical query plan. The result of applying these algebraic transformations is the logical query plan that is the output of the query-rewrite phase. The logical query plan is then converted to a physical query plan, as the optimizer makes a series of decisions about implementation of operators. The most common algebraic laws, used for simplifying expressions of all kinds, are commutative and associative laws. A commutative law about an operator says that it does not matter in which order you present the arguments of the operator; the result will be the same.

An associative law about an operator says that we may group two uses of the operator either from the left or the right [5]. Several of the operators of relational algebra are both associative and commutative. Particularly:

Commutative and associative laws for Cartesian product:

$R \times S \equiv S \times R$; $(R \times S) \times T \equiv R \times (S \times T)$;

Commutative and associative laws for joint:

$R\ \Delta\Delta\ S \equiv S\ \Delta\Delta\ R$; $(R\ \Delta\Delta\ S)\ \Delta\Delta\ T \equiv R\ \Delta\Delta\ (S\ \Delta\Delta\ T)$;

Commutative and associative laws for union:

$R \cup S = S \cup R$; $(R \cup S) \cup T = R \cup (S \cup T)$;

Commutative and associative laws for intersection:

$R \cap S = S \cap R$; $(R \cap S) \cap T = R \cap (S \cap T)$;

Splitting laws for projections:

$\pi_R (\pi_S (r)) \equiv \pi_S (r)$,

where set of R attributes is a subset of S attributes set;

Splitting laws for selection:

$\sigma_R (\sigma_S (r)) \equiv \sigma_{R \wedge S} (r)$;

Laws about selection and projection:

$\sigma_F (\pi_A (r)) \equiv \pi_A (\sigma_F (r))$;

Laws about selection and Cartesian product:

a) $\sigma_R (r_1 \times r_2) \equiv (\sigma_R (r_1)) \times r_2$,
if R contains only r_1 attributes;

b) $\sigma_R (r_1 \times r_2) \equiv r_1 \times (\sigma_R (r_2))$,
if R contains only r_2 attributes;

c) $\sigma_R (r_1 \times r_2) \equiv \sigma_{R1} (r_1) \times \sigma_{R2} (r_2)$,
R could be presented as $R_1 \wedge R_2$ if R_1 consists only of r_1 attributes, and R_2 consists only r_2 attributes;

d) $\sigma_R (r_1 \times r_2) \equiv \sigma_{R2} ((\sigma_{R1} (r_1)) \times (r_2))$,
if R_1 only consists of r_1 attributes , and r_2 consists of r_1 and r_2 attributes.

Laws about selection and union:

$\sigma_F (r_1 \cup r_2) \equiv \sigma_{F1} (r_1) \cup \sigma_{F2} (r_2)$, where r_1 and r_2 have equal relational schemas.

Laws about selection and Difference:

$\sigma_F (r_1 - r_2) \equiv \sigma_{F1} (r_1) - \sigma_{F2} (r_2)$,

where r_1 and r_2 have equal relational schemas.

Laws about projection and Cartesian product:

$\pi_A (r_1 \times r_2) \equiv \pi_B (r_1) \times \pi_C (r_2)$,

where B is an attribute in r_1 relational schema, C is an attribute of relational schema r_2, and attributes B and C are part of A.

Laws about projection and union:

$\pi_A (r_1 \cup r_2) \equiv \pi_B (r_1) \cup \pi_C (r_2)$ Where r1 and r2 have equal relational schemas, B is an attribute in r1 relational schema, C is an attribute of relational schema r2, and attributes B and C are part of A.

COMMON STRATEGIES FOR IMPROVING THE LOGICAL QUERY PLAN

When a query is converted to relational algebra, one possible logical query plan is obtained. Next step is to rewrite the plan using the algebraic laws outlined above. Alternatively, more than one logical plan could be generated by representing different orders or combinations of operations but the result to be the same. There is a single logical query plan that result ultimately in the cheapest physical plan. A query plan involving three or more relations that are arguments to the other associative and commutative operators, such as union, should be assumed to allow reordering and regrouping as the logical plan is converted to a physical plan. There are a number of algebraic laws given above that tend to improve logical query plans. The following are most commonly used in optimizers:

1. Selections should be executed as soon as possible. If a selection condition is the conjunction (AND) of several conditions, then we can split the condition and push each piece down the tree separately. This strategy is probably the most effective improvement technique.
2. Similarly, projections can be pushed down the tree, or new projections can be added.
3. Duplicate eliminations can sometimes be removed, or moved to a more convenient position in the tree.
4. Certain selections can be combined with a product below to turn the pair of operations into an equijoin, which is generally much more efficient to evaluate than are the two operations separately [5].

IMPLEMENTATION

The interactive learning tool presents the algebraic laws for optimizing query statements and visualizes the time which the query is executed for. The functional model of the system is presented in Fig. 1. The tool enables a learner to view databases and tables, and retrieve data from them. Sample database is loaded to a server in advance. For purposes of the system it should contain a large number of data records. Otherwise, the estimation of time cost will not be accurate.

The next step of the user is to execute relational operations which are listed in the figure and to see estimated execution time for each operation. The basic operations on database that could be performed are the following: Selection, Projection, Union, Difference, Cartesian product, Intersection and Joint (Natural Joint). The main functionality of learning tool is to visualize the algebraic law about operations which are given above. The equivalent expressions are applied inside system code. The purpose is to show difference between them properly. We have wanted to be sure that no mistake will appear at query formulations and students will be able to compare efficiency and effectiveness of given statements viewing the same results and different time for execution.

The objective of the implemented tool is to assist students in learning and understanding SQL queries as well as finding optimum and most efficient way to retrieve data from databases.

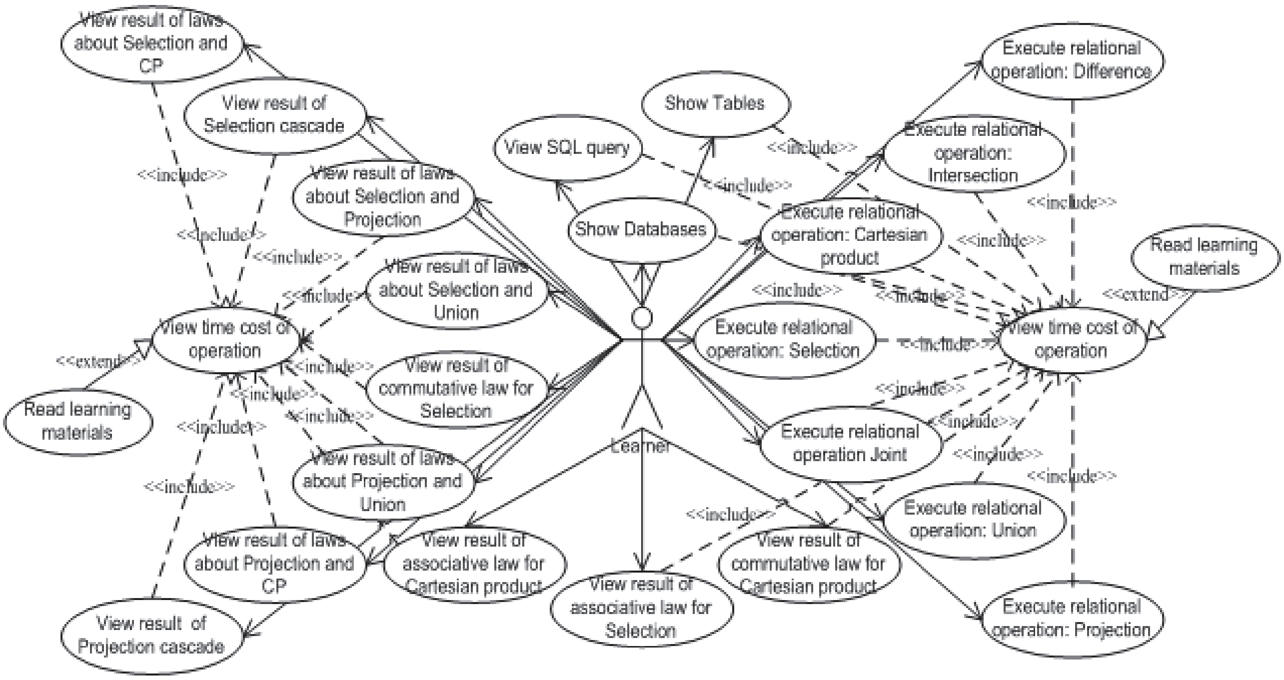

Fig.1. The functional capabilities of interactive learning tool

The interactive tool, which is web-based, will be part of an e-learning database course that will offer practical solutions of theoretical material and exercise acquired knowledge. The entire system will be used only with educational purpose, not as a commercial product.

Fig.2. Law about selection and union: $\sigma_F\ (r_1 \cup r_2)$

A user interface for visualizing law about selection and union is presented in Fig. 2 and Fig.3. The loaded expression in Fig. 2 is σ_F (r_1 U r_2) and its execution time is 0,000813 seconds, which is a little more than the time result of second expression: σF1 (r1) U σF2 (r2). Maybe the difference of times is not so considerable, because the system uses sample database with educational purpose. The effect will be substantial if database contains hundreds or million records.

Fig.3. Law about selection and union: σ_{F1} (r_1) U σ_{F2} (r_2)

CONCLUSIONS AND FUTURE WORK

In the paper, a web-based learning tool that visualizes the algebraic laws for optimizing query executions is presented. The functional capabilities of the system are explained and shown. The purpose of implemented system is to expose a SQL query execution and compare different SQL statements that give the same result considering time cost.

In the near future, we aim to perform a tool effectiveness evaluation, as well as estimate its impact on the students' learning. We also will integrate the tool into a virtual laboratory on Databases course that will consist of interactive educational modules, learning materials and tests.

REFERENCES

[1] Czejdo, B., M. Rusinkiewioz. Query transformation in an instructional database management system. Proceedings of the fifteenth SIGCSE technical symposium on Computer science education. 1984. pp: 217 – 223.

[2] De Raadt, M., S. Dekeyser, T. Y. Lee. A system employing peer review and enhanced computer assisted assessment of querying skills. Informatics in education, vol. 6 ,issue 1. January 2007, pp: 163 – 178.

[3] Query Optimization in Oracle Database10g Release 2, An Oracle White Paper, June 2005, http://www.oracle.com/technology/products/bi/db/10g/pdf/twp_general_query_optimization_10gr2_0605.pdf, Last visited 27.06.2010.

[4] Miles, William. Optimizing SQL Query Processing. http://www.kisekaeworld.com/Intractable/SQLOPT.pdf. Last visited 27.06.2010.

[5] Molina, H., J. Ullman, and J. Widom. Database Systems: The Complete Book. USA, Prentice Hall, New Jersey, 2002.

[6] Sadig, S.M., Orlowska, W., Sadig, J.Lin. SQLator: an online SQL learning workbench. ACM SIGCSE Bulletin, vol. 36 , issue 3, September 2004, pp: 223 – 227.

ABOUT THE AUTHORS

Elitsa Arsova, PhD Student, Dept. of Computer Systems and Technologies, University of Ruse, Phone: +359 (0)82 888 827, E-mail: EArsova@ecs.uni-ruse.bg

Assist. Prof. Silyan Arsov, PhD, Dept. of Computer Systems and Technologies, University of Ruse, Phone: +359 (0)82 888 276, E-mail: SArsov@ecs.uni-ruse.bg